01 Jan 2005

Gene Regulatory Networks Inference with Recurrent Neural Network Models

Rui Xu
Missouri University of Science and Technology

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Gene Regulatory Networks Inference with Recurrent Neural Network Models

Rui Xu and Donald C. Wunsch II
Applied Computational Intelligence Laboratory
Dept. of Electrical and Computer Engineering
University of Missouri - Rolla
Rolla, MO 65409-0249 USA
rxu@umr.edu, dwunsch@umr.edu

Abstract – Large-scale time series gene expression data generated from DNA microarray experiments provide us a new means to reveal fundamental cellular processes, investigate functions of genes, and understand their relations and interactions. To infer gene regulatory networks from these data with effective computational tools has attracted intensive efforts from artificial intelligence and machine learning. Here, we use a recurrent neural network (RNN), trained with particle swarm optimization (PSO), to investigate the behaviors of regulatory networks. The experimental results, on a synthetic data set and a real data set, show that the proposed model and algorithm can effectively capture the dynamics of the gene expression time series and are capable of revealing regulatory interactions between genes.

I INTRODUCTION

With the rapid advancement of DNA microarray technologies [1, 2], inferring genetic regulatory networks from time series gene expression data has become increasingly important, in order to reveal fundamental cellular processes, investigate functions of genes, and understand complex relations and interactions among genes [3, 4]. A genetic regulatory network consists of a set of DNA, RNA, proteins, and other molecules, and describes regulatory mechanisms among these components. Genetic information that determines structures, functions, and properties of living cells is stored in DNA, whose coding regions, known as genes, encode proteins. According to the central dogma of molecular biology, genes are transcribed into mRNA molecules, which are then translated to proteins. Since all cells for a specific organism include the same genetic material, it is important to know which proteins are synthesized, or which genes are expressed, under certain conditions. This is achieved through the actions of some proteins, which can activate or inhibit the transcription rate of certain genes by binding to their regulatory regions. Therefore, the transcription of a specific gene, or the control of its gene expression, can be regarded as a combinatorial effect of a set of other genes.

Conventional methods can only investigate activities between a pair of genes, or among several genes, which is far from sufficient, for exploring the complicated regulatory mechanisms. DNA microarray technologies provide an effective and efficient way to measure gene expression levels of thousands of genes simultaneously under different conditions, which makes it possible to investigate gene activities from the angle of the whole genome [3]. Several computational models, rooted in artificial intelligence and machine learning, have been proposed to unveil the behaviors of regulatory networks from time series gene expression data [4].

Boolean networks are binary models, which consider that a gene has only two states: 1 for active and 0 for inactive [5-7]. The effect of other genes on the state change of a given gene is described through a Boolean function. Although Boolean networks make it possible to explore the dynamics of a genetic regulatory system, they ignore the effect of genes at intermediate levels. Loss of information is inevitable with the discretizaiton process. Furthermore, Boolean networks assume the transitions between activation states of the genes are synchronous, which usually is not true. Bayesian networks are graph models that estimate complicated multivariate joint probability distributions through local probabilities [8]. Under this framework, a genetic regulatory network is described as a directed acyclic graph, including a set of vertices and edges. The vertices are related to random variables and represent genes or other components while the edges capture the conditional dependence relation and represent the interactions among genes. Bayesian networks are effective in dealing with noise, incompleteness, and stochastic aspects of gene expression data. Graph representation makes it convenient to investigate interactions between the genes. However, Bayesian networks do not consider dynamical aspects of gene regulation and leave temporal information unhandled. Recently, dynamic Bayesian networks (DBN) attract more attention [9-11], DBN can model behaviors emerging temporally, and effectively handle problems like hidden variables, prior knowledge, and missing data. The disadvantage of DBN is that they cannot scale well to large-scale data sets. For the linear additive regulation models [12-14], the expression level of a gene at a certain time point can be calculated by the weighted sum of the expression levels of all genes in the network at a previous time point. Although linear additive regulation can reveal certain linear relations in the regulatory systems, it lacks the

0-7803-9048-2/05/$20.00 ©2005 IEEE

capability to capture the nonlinear dynamics between gene regulations.

Considering the limitations of these methods, here, we use recurrent neural network (RNN) models to infer gene regulatory networks from time series gene expression data. In using RNNs for gene network inference, we are mainly concerned with the ability of RNNs to interpret complex temporal behavior. Generalized recurrent neural network models can be considered as signal processing units forming a global regulatory network. The similarity between RNNs and gene networks makes RNNs an important method in unraveling the mystery of gene regulation relationships. In order to estimate the networks parameters, we use particle swarm optimization (PSO) as the training algorithm, which is an evolutionary computational algorithm for global optimization, based on the simulation of complex social behavior [15-16]. The effectiveness of the model is demonstrated by the simulation on a synthetic data set and a real data set.

The paper is organized as follows. Section II describes the model and training algorithm of recurrent neural networks used for regulatory network inference. Experimental results on the synthetic and real data sets are summarized in Section III and section IV concludes the paper.

II. RECURRENT NEURAL NETWORKS

A. Model

For a continuous time system, the genetic regulation model can be represented through a recurrent neural network formulation [17-20],

$$\tau_i \frac{de_i}{dt} = f(\sum_{j=1}^{N} w_{ij} e_j + \sum_{k=1}^{K} v_{ik} u_k + \beta_i) - \lambda_i e_i \qquad (1),$$

where e_i is the gene expression level for the i^{th} gene ($1 \le i \le N$, N is the number of genes in the model), $f()$ is a nonlinear function (usually, sigmoid function is used $f(z) = 1/(1+e^{-z})$), w_{ij} represents the effect of the j^{th} gene on the i^{th} gene ($1 \le i, j \le N$), u_k is the k^{th} ($1 \le k \le K$, K is the number of external variables) external variable, v_{ik} represents the effect of the k^{th} external variable on the i^{th} gene, τ is the time constant, β is the bias term, and λ is the decay rate parameter. A negative value of w_{ij} represents the inhibition of the j^{th} gene on the i^{th} gene, while a positive value indicates the activation controls. When w_{ij} is zero, there is no influence of the j^{th} gene on the expression change of the i^{th} gene.

Correspondingly, its discrete form is

$$e_i(t+\Delta t) = \frac{\Delta t}{\tau_i} f(\sum_{j=1}^{N} w_{ij} e_j(t) + \sum_{k=1}^{K} v_{ik} u_k(t) + \beta_i) + \left(1 - \frac{\lambda_i \Delta t}{\tau_i}\right) e_i(t) \qquad (2).$$

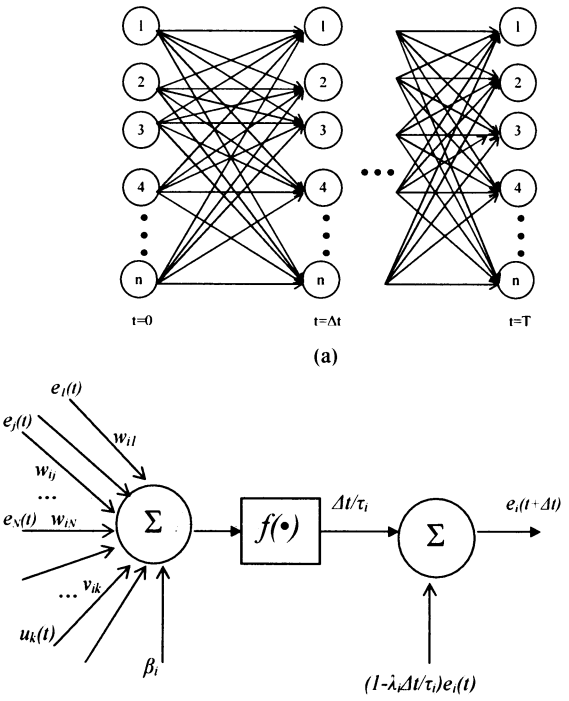

Fig. 1. (a) The description of a genetic network through a recurrent neural network model. This network is unrolled in time from $t=0$ to T with an interval Δt. Here, the regulatory network is shown in a fully connected form, although, in practice, the network is usually sparsely connected. (b) A node (neuron) in the recurrent neural network model

Fig. 1 (a) depicts a recurrent neural network, which is unrolled in time from $t=0$ to T with an interval Δt, for modeling genetic network, Here, each node corresponds to a gene and a connection between two nodes defines their interaction. The weight values can be either positive, negative, or zero, as mentioned above. Fig. 1 (b) illustrates a node in the recurrent neural network, which realizes the equation in (2).

Since it is usually difficult to have the measurements of the external variables, it is a common practice to ignore the term $\sum_{k=1}^{K} v_{ik} u_k(t)$. From the following section, we can see that its addition does not affect the derivation of the learning algorithm. In our work, we also assume that the decay rate parameter λ is 1. The final model we process in the paper is represented as

$$e_i(t+\Delta t) = \frac{\Delta t}{\tau_i} \times f(\sum_{j=1}^{N} w_{ij} e_j(t) + \beta_i) + \left(1 - \frac{\Delta t}{\tau_i}\right) e_i(t) \qquad (3).$$

B. Training algorithm

There exist ample algorithms for RNN training in the literature, e.g., back-propagation through time (BPTT) and genetic algorithm (GA). BPTT was first proposed by Paul Werbos [21], as an extension of the standard back-propagation algorithm. By using BPTT, we find the

derivatives of the cost function with respect to the individual weights of the network. These derivatives can be used to do gradient descent on the weights, updating them in the direction that minimizes the error [22]. However, the use of gradient descent required the error function to be differentiable, and also makes the procedure easy to get stuck in some local minima. We used BPTT to train the RNN models for genetic networks inference, and the results are reported in [23] and [24]. Inspired by the natural evolution process, GA tends to optimize a population of structure, evaluated according to a fitness function, by using a set of evolutionary operators [25].

Here, we use particle swarm optimization (PSO) [16] for network parameters learning. Similar to GA, PSO is based on a swarm of particles, each of which represents a candidate solution in the multidimensional problem space. The difference lies that a random velocity is associated with each particle, which is considered to "be flown through the problem space" [16]. The basic idea of PSO is to accelerate each particle towards its previous best solution, called *pbest*, and the overall best locations in the swarm, called *gbest*, at each time step. These best locations are determined based on the calculated values of the defined fitness function. This concept is depicted in Fig. 2, in which L^t and L^{t+1} represent the locations at current and next time point, V^t and V^{t+1} represent the velocities at current and next time point, V_{pbest} is the velocity according to *pbest*, and V_{gbest} is the velocity according to *gbest*. It has been shown that PSO require less computational cost and can achieve faster convergence than conventional back-propagation in training feedforward neural networks for approximating a nonlinear function [26]. Meanwhile, compared with GA, PSO has many desirable characteristics, e.g., the flexibility in balancing global and local search, computational efficiency on both time and memory, and the ease to implement. Particularly, PSO is equipped with the memory mechanism for keeping previous best solutions, therefore, avoids the possible loss of learned knowledge. PSO proves to be a powerful tool to explore sophisticated space [15], and this makes it suitable for regulatory networks inference.

Given a set of particles $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M)$, where M is the number of particles in the swarm, the i^{th} particle (candidate solution) can be represented as a D-dimensional vector $\mathbf{x}_i = (w_{i,11}, ..., w_{i,N1}, w_{i,12}, ..., w_{i,1N}, ..., w_{i,NN}, \beta_{i,1}, ..., \beta_{i,N}, \tau_{i,1}, ..., \tau_{i,N})$, $1 \le i \le M$, where $D = N(N+2)$. Its velocity is described as $\mathbf{v}_i = (v_{i,1}, v_{i,2}, ..., v_{i,D})$. A fitness function is used to measure the deviation of network output $e(t)$ from the real measurement (target) $d(t)$, defined as

$$Fitness(x_i) = \frac{1}{TN} \sum_{t=0}^{T} \sum_{i=1}^{N} (e_i(t) - d_i(t))^2 \qquad (4).$$

More elaborate error terms can be easily added based on the specific requirement of the problem at hand. Note here,

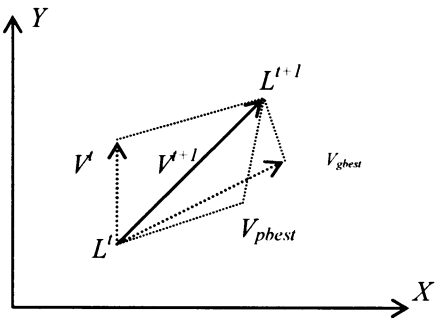

Fig.2. Basic concept of the position change of a particle in PSO [27].

we use a batch mode for training, which means the parameters updating is performed after all input data points are presented to the model [22, 26]. The basic procedure of implementing PSO for learning network parameters in the RNN model can be summarized as follows.

i). Initialize a population of particles with random positions and velocities of D dimensions. Specifically, the connection weights, biases, and time constants are randomly generated with uniform probabilities over the range $[w_{\min}, w_{\max}]$, $[\beta_{\min}, \beta_{\max}]$, and $[\tau_{\min}, \tau_{\max}]$, respectively. Similarly, the velocities are randomly generated with uniform probabilities in the range $[-V_{\max}, V_{\max}]$, where $V_{\max}$ is the maximum value of the velocity allowed.

ii). Calculate the estimated gene expression time series based on the RNN model and evaluate the optimization fitness function for each particle.

iii). Compare the fitness value of each particle with its *pbest*. If current value is better than *pbest*, reset both *pbest* value and location to the current value and location.

iv). Compare the fitness value of each particle with *gbest*. If current value is better than *gbest*, reset *gbest* to the current particle's array index and value.

v). Update the velocity and position of the particles with the following equations

$$\mathbf{v}_i = W_I \times \mathbf{v}_i + c_1 \times rand_1 \times (\mathbf{pbest}_i - \mathbf{x}_i) + c_2 \times rand_2 \times (\mathbf{gbest}_i - \mathbf{x}_i) \qquad (5)$$

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i \quad (1 \le i \le M) \qquad (6)$$

where W_I is the inertia weight, c_1 and c_2 are the acceleration constants, and $rand_1$ and $rand_2$ are uniform random functions.

vi). Return to step ii until a stop criterion is satisfied. Usually the learning stops when a maximum number of iterations or high-quality solutions are met.

PSO has only four major parameters needed to be determined in advance. The inertial weight W_I is designed as a tradeoff between the global and local search. Larger values of W_I facilitate global exploration while lower values encourage local search. Usually, the inertial weight is set to

decrease linearly. Here, we utilize a strategy that generates a random value varying between 0 and W_{max}, represented as $W_I = W_{\max} - (rand/2)$, where *rand* is a uniform random function. In our study, this strategy usually achieves better result than any other methods. c_1 and c_2 are known as the cognition and social components, respectively, and are used to control the effects of a particle itself and its surrounding environment, which is achieved through adjusting the velocity towards *pbest* and *gbest*. Commonly, both parameters are set to 2.0 based on the past experience [15]. During the evolutionary procedure, the velocity for each particle is restricted to a limit $V_{\max}$, like the way in velocity initialization. When the velocity exceeds $V_{\max}$, it is re-assigned to $V_{\max}$. If $V_{\max}$ is too small, particles may become trapped into local optima, while if $V_{\max}$ is too large, particles may miss some good solutions. $V_{\max}$ is usually set around 10-20% of the dynamic range of the variable on each dimension [15].

III. RESULTS

We applied the recurrent neural network model, along with the PSO training algorithm, to a synthetic data set and a real data set. The goal is to recover the basic genetic regulatory networks from the generated time series gene expression data.

The interaction weight matrix W, the bias β, and the time constant τ for the simplified genetic network, which has 4 genes and was used in [28], are set as in Table I. The total number of weights that are non-zero is 8, which is half of the total number of weights. The network was simulated from a random initial state for each gene. We generated 300 time points based on Equation (3), at a time resolution of Δt=0.1. The expression levels for these genes generally get saturated very fast, since we do not consider the stimulus from the external environment. We performed analysis for both single time series and multiple (including three series) time series, in which 100 time points are used for each time series and most of them were taken from the early stage of the process.

We ran the algorithm 300 times with different random initial values for the weights, biases, and time constants. The performance is based on the averages across these experiments, unless otherwise indicated. This is achieved by checking the *gbest* solution for each run. The network weights were evolved for 10,000 generations. The parameters for PSO are set as follows: W_{max}=0.7, $c_1 = c_2 = 2$, and $V_{\max} = 2$. The initial values for the weights (including biases) and time constants lie between -1 and 1 and 0 and 10, respectively.

One of the major obstacles for current exploration is "curse of dimensionality", which indicates the exponential growth in computational complexity and the demand for more samples as a result of high dimensionality in the feature space [14, 22, 29]. Typically, gene expression data contain measurements of thousands of genes, but only have a limited number of time points. This situation limits the application of many data-driven computational models and makes it very difficult to infer a fully determined large-scale regulatory network. Fortunately, in genetics, it is assumed that a gene is only regulated by a limited number of genes. In other words, the regulatory networks are sparsely connected rather than fully connected. In this sense, it is reasonable to identify the weights whose values are non-zeroes from expression data, which indicate the potential interactions between genes, and furthermore, whether the interaction is activation or inhibition, based on the sign of the weights, although it may not be possible to accurately recover the values of the weights, due to the limitation of the available time points. Based on the results of the 300 runs, we discretize the weights into three classes according to the criterion similar to what are used in [9]:

- class + representing activation: $\mu_{ij} > \mu + \sigma$ and $\sigma_{ij} < |\mu_{ij}|$;
- class – representing inhibition: $\mu_{ij} < \mu - \sigma$ and $\sigma_{ij} < |\mu_{ij}|$;
- class 0 representing absence of regulation: otherwise.

where μ_{ij} and σ_{ij}^2 are the mean and variance for the element w_{ij} in the weight connection matrix and μ and σ^2 are the mean and variance of the means of all the 16 elements. The original and the identified weight connection matrix with the single series and multiple series are summarized in Table II. Compared with the original weight matrix, both ways can find some important relations existing in the network. For the single time series, 4 out of 8 non-zero weights are correctly identified in the inferred genetic network, while for three time series, 5 out of 8 non-zero weights are identified with only one false positive, which is wrongly identified as activation between genes. By using more time series, more information is provided to the model, therefore, we can usually achieve better result, which agrees with the conclusion in [20].

We also employed the proposed method to analyze the SOS DNA Repair network in bacterium *Escherichia coli*

TABLE I
THE SYNTHETIC GENETIC NETWORK USED FOR THE GENERATION OF THE DATA

w_{IJ}				β_I	τ_I
20.0	-20.0	0.0	0.0	0.0	10.0
15.0	-10.0	0.0	0.0	-5.0	5.0
0.0	-8.0	12.0	0.0	0.0	5.0
0.0	0.0	8.0	-12.0	0.0	5.0

TABLE II
THE GENERATED CONNECTION MATIRX (UPPER PANEL) AND THE LEARNED CONNECTION MATRIX WITH THE SINGLE SERIES (SECOND PANEL) AND MULTIPLE SERIES (LOWER PANEL). EACH ELEMENT w_{ij} IN THE MATRIX REPRESNETS THE RELATION BETWEEN THE i^{th} AND j^{th} GENE, AS ACTIVATION (+), INHIBITION (-), AND ABSENCE OF REGULATION (0).

w_{IJ}			
+	-	0	0
+	-	0	0
0	-	+	0
0	0	+	-
+	-	+	0
0	0	0	0
+	-	0	0
0	0	0	-
+	-	0	0
0	0	0	0
+	-	+	0
0	0	0	-

depicted in Fig. 3 [31]. When damage occurs, protein RecA, which functions as a sensor of DNA damage in the SOS system, becomes activated and mediates LexA autocleavage by binding to single-stranded DNA molecule. Protein LexA is a master repressor that represses all genes when no damage occurs. The drop in LexA expression levels causes the activation of the SOS genes. After the damage is repaired or bypassed, the expression level of RecA drops, which causes the accumulation of LexA. Then, LexA binds sites in the promoter regions of these SOS genes and represses their expression. The cells return to their original states. Four experiments have been conducted by Uri Alon with different light intensities and each experiment includes the expression measurements for 8 major genes (uvrD, lexA, umuD, recA, uvrA, uvrY, ruvA, and polB) through 50 time points, sampled every 6 minutes. In our study, we only used the data from experiment 2 for further analysis.

We set the major parameters for PSO as before. Fig. 4 shows the real gene expression profiles and the learned mean expression profiles with PSO. The average mean square error between the real profiles and learned profiles is 0.0313. We

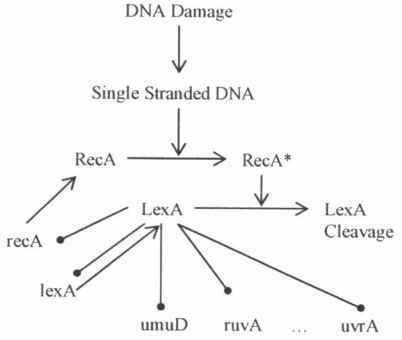

Fig. 3. The SOS DNA Repair network [31]. Inhibitions are represented by -●, while activations are represented by →.

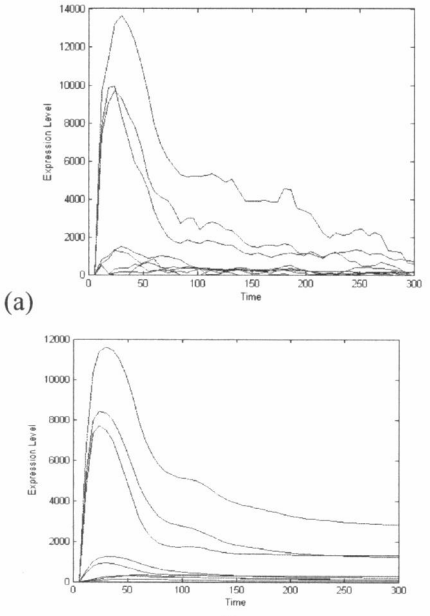

Fig. 4. (a) The measured gene expression profiles for Exp. 2; (b) The learned mean expression profiles with PSO (300 runs).

can see the proposed model can effectively capture the dynamics of most genes (lexA, recA, uvrA, uvrD, and umuD) in the network, and the major change trends of the gene expression levels are reflected in the learning curves. The expression profiles for gene uvrY, ruvA, and polB oscillate dramatically between the maximum value and zero, and the obtained models generally use their means to represent the profiles because of the definition of the fitness function. However, when we tried to infer the potential interactions among genes by using the criterion aforementioned, we find that we can only correctly identify 2 out of 9 potential connections between genes. These connections we find include the inhibition of lexA on uvrD, and the activation of recA on lexA. The variance is quite large for most of the weights, which causes most of the weights are regarded as zeros. Also, three false positives are included, which do not exist in the network. The result may suggest that, although we can find some meaningful insight between the genes by using this simple criterion, it is still too rough to be used to deal with more complicated genetic networks satisfactorily. We propose an iterative procedure with PSO to select important non-zero weights [30] and the information about motif can also be used to examine the validity of the unveiled relations in the genes [10]. This method can achieve better results in analyzing larger genetic networks; however, it is still work-in-progress. [30].

IV. CONCLUSIONS

To understand the genetic regulatory mechanisms is one of the central tasks in molecular genetics. Inference of genetic regulatory networks based on the time series gene expression

data from microarray experiments becomes an important and effective way to achieve this goal. Here, we utilize recurrent neural network models to model regulatory systems and unveil potential gene interactions. Initial experiments have shown some promising results and demonstrate the potential of this model in gene regulatory network inference. However, with the limited data, current research only focuses on the modeling of network from synthetic data, or simulation of small-scale network including only several genes or gene clusters. Several strategies, such as clustering and interpolation, have been proposed to deal with this still-open problem. No attempt has been seen to infer large-scale genetic regulatory networks. High quality time series gene expression data with sufficient number of time points is particularly important. In the meantime, further improvement for the current computational models is also required in order to explore gene regulation more effectively. Another factor that needs further investigation is the time delay, which is ubiquitous in gene regulatory activities and has already been widely reported in the literature. To incorporate time delay into the current model may reflect the system dynamics more effectively. Furthermore, genetic networks are known to be robust to noise. Gene expression levels in the networks will not be affected greatly due to the small changes, caused by noise, in expression levels of some genes. Many genes have same functions and express themselves in a similar way under certain pathways. Proposed computational models should be capable of interpreting these phenomena.

Acknowledgment

Partial support for this research from the National Science Foundation, and from the M.K. Finley Missouri endowment, is gratefully acknowledged.

References

[1] M. Eisen and P. Brown, "DNA Arrays for Analysis of Gene Expression," *Methods Enzymol*, vol. 303, pp. 179-205, 1999.

[2] R. Lipshutz, S. Fodor, T. Gingeras, and D. Lockhart, "High Density Synthetic Oligonucleotide Arrays," *Nature Genetics*, vol. 21, pp. 20-24, 1999.

[3] P. D'haeseleer, S. Liang, and R. Somogyi, "Genetic Network Inference: From Co-expression Clustering to Reverse Engineering," *Bioinformatics*, vol. 16, no. 8, pp. 707--726, 2000.

[4] H. De Jong, "Modeling and Simulation of Genetic Regulatory Systems: A Literature Review", *Journal of Computational Biology*, vol. 9, pp. 67-103, 2002.

[5] S. Kauffman, "The Origins of Order: Self-Organization and Selection in Evolution", Oxford University Press, New York, 1993.

[6] S. Liang, S. Fuhrman, and R. Somogyi, "REVEAL: A General Reverse Engineering Algorithm for Inference of Genetic Network Architectures", *Proceedings of the Pacific Symposium Biocomputing (PSB'98)*, vol. 3, pp. 18-29, 1998.

[7] I. Shmulevich, E. Dougherty, and W. Zhang, "From Boolean to Probabilistic Boolean Networks as Models of Genetic Regulatory Networks", *Proceedings of IEEE*, vol. 90, no. 11, pp. 1778- 1792.

[8] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian Networks to Analyze Expression data", *Journal of Computational Biology*, vol. 7, pp. 601-620, 2000.

[9] B. Perrin, L. Ralaivola, A. Mazurie, S. Battani, J. Mallet, and F. d'Alché-Buc, "Gene Networks Inference Using Dynamic Bayesian Networks", *Bioinformatics*, vol. 19, Suppl.2, pp. ii138- ii148, 2003.

[10] Y. Tamada, S. Kim, H.Bannai, S. Imoto, K. Tashiro, S. Kuhara, and S. Miyano, "Estimating Gene Networks from Gene Expression Data by Combing Bayesian Network Model with Promoter Element Detection", *Bioinformatics*, vol. 19, Suppl.2, pp. ii227- ii236, 2003.

[11] D. Husmeier, "Sensitivity and Specificity of Inferring Genetic Regulatory Interactions from Microarray Experiments with Dynamic Bayesian Networks", *Bioinformatics*, vol. 19, no. 17, pp. 2271- 2282, 2003.

[12] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi, "Linear Modeling of mRNA Expression Levels during CNS Development and Injury", In R.B. Altman, K. Lauderdale, A.K. Dunker, L. Hunter, and T.E. Klein, editors, *Proceedings of the Pacific Symposium Biocomputing (PSB'99)*, pp. 41-52., 1999.

[13] E. van Someren, L. Wessels, M. Reinders, "Linear Modeling of Genetic Networks from Experimental Data", *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB00)*, pp. 355-366, 2000.

[14] P. D'haeseleer, "Reconstructing Gene Network from Large Scale Gene Expression Data", Dissertation, University of New Mexico, 2000.

[15] R. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and recourses," *Proc. of the 2001 Congress on Evolutionary Computation*, vol. 1, pp. 81-86, 2001.

[16] J. Kennedy, R. Eberhart, Y. Shi, "Swarm Intelligence", Morgan Kaufmann Publishers, 2001.

[17] E. Mjolsness, T. Mann, R. Castaño, and B. Wold, "From Co-expression to Co-regulation: An Approach to Inferring Transcriptional Regulation among Gene Classes from Large-scale Expression Data", *in Advances in neural Information Processing Systems 12*, pp. 928-934, MIT Press, 2000.

[18] D. Weaver, C. Workman, and G. Stormo, "Modeling Regulatory Networks with Weight Matrices", *Proceedings of the Pacific Symposium on Biocomputing*, pp. 112-123, 1999.

[19] J. Vohradsky, "Neural Network Model of Gene Expression", *the FASEB Journal*, vol. 15, pp. 846-854, 2001.

[20] M. Wahde and J. Hertz, "Modeling Genetic Regulatory Dynamics in Neural Development", *Journal of Computational Biology*, 8, 429-442, 2001.

[21] P.J. Werbos, "Backpropagation Through Time: What It Does And How to Do It", *Proceedings of IEEE*, 78(10), pp. 1550-1560, 1990.

[22] S. Haykin, "Neural Networks: A Comprehensive Foundation", *2nd Ed.*, Prentice Hall, New Jersey, 1999.

[23] Rui Xu, Xiao Hu, and Donald C. Wunsch II, "Inference of Genetic Regulatory Networks with Recurrent Neural Network Models," *Proceedings of the 26th International Conference of IEEE EMBS*, September, 2004.

[24] X. Hu, A. Maglia, and D. Wunsch II, "A General Recurrent Network Approach to Model Decay Constants in A Genetic Network", submitted for publication 2004.

[25] D. Fogel, "An Introduction to Simulated Evolutionary Optimization," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3-14, 1994.

[26] V. Gudise and G. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks", *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 110-117, 2003.

[27] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1232-1239, 2000.

[28] M. Wahde and J. Hertz, "Coarse-grained Reverse Engineering of Genetic Regulatory Networks", *Biosystems*, 55, 129–136, 2000.

[29] E. van Someren, L. Wessels, and M. Reinders, "Genetic Network Models: A Comparative Study", *In Proc. of SPIE, Micro-arrays: Optical Technologies and Informatics (BIOS01)*, vol. 4266, pp. 236--247, 2001.

[30] R. Xu and D. Wunsch, "Genetic Regulatory Networks Inference with Particle Swarm Optimization," in preparation, 2005.

[31] M. Ronen, R. Rosenberg, B. Shraiman, and U. Alon, "Assigning Numbers to the Arrows: Parameterizing A Gene Regulation Network by Using Accurate Expression Kinetics", *Proc. Natl. Acad. Sci.*, vol. 99, no. 16, pp.10555-10560, 2002.